CHALMERS
UNIVERSITY OF TECHNOLOGY

Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems

Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems

Wissam Aoudi
Chalmers University of Technology
Gothenburg, Sweden
wissam.aoudi@chalmers.se

Mikel Iturbe
Mondragon University
Arrasate-Mondragón, Spain
miturbe@mondragon.edu

Magnus Almgren
Chalmers University of Technology
Gothenburg, Sweden
magnus.almgren@chalmers.se

ABSTRACT

Recent incidents have shown that Industrial Control Systems (ICS) are becoming increasingly susceptible to sophisticated and targeted attacks initiated by adversaries with high motivation, domain knowledge, and resources. Although traditional security mechanisms can be implemented at the IT-infrastructure level of such cyber-physical systems, the community has acknowledged that it is imperative to also monitor the process-level activity, as attacks on ICS may very well influence the physical process. In this paper, we present PASAD, a novel stealthy-attack detection mechanism that monitors time series of sensor measurements in real time for structural changes in the process behavior. We demonstrate the effectiveness of our approach through simulations and experiments on data from real systems. Experimental results show that PASAD is capable of detecting not only significant deviations in the process behavior, but also subtle attack-indicating changes, significantly raising the bar for strategic adversaries who may attempt to maintain their malicious manipulation within the noise level.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**;

KEYWORDS

Intrusion Detection; Industrial Control Systems; Singular Spectrum Analysis; Stealthy Attacks; Cyber-Physical Systems; Isometry Trick; Partial Isometry; Departure Detection

ACM Reference Format:
Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. 2018. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3243734.3243781

1 INTRODUCTION

Industrial Control Systems (ICS) are often found in critical infrastructures, such as transportation, aerospace, electricity grids, nuclear plants, and gas distribution systems to name a few. Unlike traditional IT systems, which mainly manage data, ICS control physical processes. The need to secure these critical cyber-physical systems cannot be overemphasized as the impact of cyber attacks is no longer bounded by financial losses due to some service disruption or loss of data. Cyber attacks on control systems can cause irreparable physical damage to equipment in safety-critical facilities, raw sewage to spill out into local parks and rivers, large-scale power blackouts, and severe damage to a nation's critical assets on which normal societal functioning depends. In recent years, several incidents have been reported indicating that ICS are becoming increasingly exposed to *targeted* attacks, allegedly initiated by adversaries with enough skills and resources to circumvent security measures at the IT-infrastructure level and trigger unwanted behaviors in the underlying physical process. Stuxnet [10, 14], the German steel-mill attack [33], the Maroochy water breach [2], Triton [25], and the growing number of attacks on energy networks [34, 53] are just a few cases in point. This issue is even being addressed in high-level forums such as in the 2018 World Economic Forum meeting in Davos, where concerns have been raised about the consequences of successful attacks on water supply or nuclear power stations, which have potential to claim many lives and throw communities into chaos [3].

Due to the high rewards that the attacks on ICS can realize, these systems are becoming attractive targets for cyber criminals. Consequently, ICS-specific malware and targeted attacks are growing both in diversity and sophistication. In addition to the prominent Stuxnet worm, other pieces of malware designed specifically for ICS, such as the Dragonfly malware [41], and attacks specifically targeting Programmable Logic Controllers (PLCs), including the PLC-Blaster worm [47], the Ladder Logic Bombs [20], and the PLC Pin Control attack [1] have started to surface.

Evolving from isolated systems running proprietary control protocols using specialized hardware and software, ICS are increasingly adopting IT solutions by using industry-standard network protocols and operating systems to promote corporate connectivity and meet performance requirements [48]. The integration of standard IT-based solutions is making these systems considerably less isolated from the outside world, and introducing a host of new vulnerabilities inherited from the IT sector. According to a 2017 SANS survey [21], 69% of security practitioners perceive the current cyber-threat level in ICS as critical or high, a whooping 26% increase from just two years before.

One fundamental difference between traditional IT-based systems and industrial control systems is that the latter interact with the physical world. Conventional off-the-shelf intrusion detection systems prove ill-fit for the ICS domain because they do not take process semantics into account [22]. Reportedly, in the latest attack on the power grid in Ukraine, hackers are thought to have hidden, *undetected for six months*, in the energy company's IT network,

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3243781

acquiring privileges to access systems before taking methodical steps to take the power offline [44]. In the attack on the Iranian nuclear plant, the Stuxnet worm used zero-day exploits to infect systems while hiding its changes using sophisticated rootkits and validating its drivers with stolen trusted certificates [8, 14]. The fundamentally different nature of these systems calls for security approaches that are tailored to the ICS environment.

There is an emerging trend in ICS intrusion detection research where researchers have shown growing interest in developing techniques that can detect such sophisticated attacks at the **process level** [8, 9, 22, 28, 31, 35, 46, 50, 51]. Although intruders may be able to hide in the IT network while figuring out their workings, the main driver of this trend is the observation that they can hardly hide their final goal, which is *to cause an adverse effect on the physical process by maliciously manipulating sensor and control data* [8].

A process-level intrusion detection system monitors sensors—the eyes and ears of control systems—and possibly control commands, to determine if the physical process is drifting from the normal or expected behavior. One popular approach frequently used in this domain proposes to build a Linear Dynamical State-Space (LDS) model of the physical process, through what is known as system identification (see [51] and references therein), which is subsequently used to detect anomalies in the system behavior. Although such approaches might detect anomalous behaviors, models are difficult to build, requiring massive human effort at the preliminary stage [15], and a complete and highly detailed model of the physical process that is not always available [28].

In this paper, we propose a *specification-agnostic* approach that is purely data-driven and requires no prior knowledge of the system dynamics. We present **PASAD**, a light-weight and fast model-free Process-Aware Stealthy-Attack Detection mechanism that monitors sensors in ICS in real time and raises an alarm whenever a *structural change* in the behavior of the physical process is suspected.

A data-driven approach has also been proposed in [22], where historical sensor readings are used to fit a linear Auto-Regressive (AR) model, which is then used to detect deviations from what the model expects. We argue that attempting to fit a simple linear model to a time series of noisy sensor measurements may often yield inaccurate detection results when the changes in the time series are subtle. In §4.4, we present a comparison between AR and PASAD and show that our method performs better.

Most of the existing approaches use prediction-based methods where a predicted value $\hat{y}_k$ of the sensor reading y_k at time k is computed, and the difference $r_k = |\hat{y}_k - y_k|$ is checked against a threshold θ such that if $r_k \geq \theta$, an alarm is raised. *Instead of predicting the future, PASAD seeks to solve the easier problem of deciding whether present sensor readings are departing from past readings due to a change in the mechanism generating them.* We review related literature in more detail in §5.

In addition to the aforementioned benefits, what we consider the chief advantage of our approach over existing methods is the fact that PASAD is capable of detecting *slight variations* in the sensor signal, owing to its impressive noise-reduction capabilities. This leads to the possibility of detecting *strategic attackers who may try to hide their attacks even at the process level*, by injecting just enough false data that the compromised sensor values remain roughly within the noise level. Such **stealthy** integrity attacks are hard to detect by failure detectors or anomaly detectors that are not insensitive to noise. Mo and Sinopoli [38] argue that a strategic adversary may inject an attack that inflicts a large perturbation on the system state while only causing a slight increase in the detection rate of these detectors. We further motivate our approach in §2.1.

PASAD initially extracts noise-reduced signal information from a time series of sensor measurements during normal process operation and then actively checks whether present realizations of the process are *departing* from historical normal behavior. To extract signal information, PASAD borrows ideas from **Singular Spectrum Analysis (SSA)** [7, 13, 17–19, 23, 52], a non-parametric exploratory analysis tool for time series that is particularly suitable for separating the *deterministic* part of a dynamical system behavior from the *chaotic* part, purely from noisy time series of measurements.

Once signal information has been extracted, PASAD proceeds by identifying a **signal subspace** that describes the deterministic variability in the time series produced by the process during normal activity. Afterwards, the most recent observations (lagged vectors thereof) are projected onto this subspace and a *departure score* is computed for every new observation. A persistent increase in the computed score suggests that current observations are not in accordance with the estimated dynamics, and that a malicious change in the mechanism generating the time series may be occurring.

Our method is based on a rich and sound theory and at the same time enjoys a low computational overhead. A key ingredient to this desired combination is a direct result of what we refer to as the **isometry trick**. The trick is based on a mathematical property and has two added benefits for PASAD: one pertaining to *efficiency*, where the computations needed to evaluate the departure score are reduced significantly; and another pertaining to *validation*, where one can visualize the time-series data in the signal subspace, which helps validating the underlying theory. We dedicate §2.6 to explaining and proving this fundamental property.

We demonstrate the effectiveness of PASAD using the Tennessee-Eastman (TE) process control model under various attack scenarios. While it is shown how direct attacks that aim to sabotage the control system by forcing controllers to operate outside their specified boundaries can be trivially detected, we demonstrate how stealthy integrity attacks, which are designed to cause tangible impact in a stealthy manner, may as well be detected in reasonable time. Furthermore, we test PASAD using data from the Secure Water Treatment (SWaT) plant—a physical testbed dedicated for ICS security research—and on data from a relatively long network trace captured from an operational water distribution plant in Sweden, to investigate its applicability to real industrial settings. Finally, we compare our method with an enhanced version of the AR method proposed in [22], and demonstrate how, in contrast to simple linear models like AR, PASAD is capable of detecting subtle attack-indicating changes in the process behavior.

More specifically, our contributions in this paper are the following: *(i)* We present a novel technique for process-level detection of stealthy attacks on control systems that is capable of detecting strategic attackers who may attempt to camouflage malicious changes with noise. We provide theoretical arguments and empirical results validating our principled approach; *(ii)* we introduce the notion of *departure* as a specific type of anomaly, whereby a process regulated by some control system departs from the normal

state due to a structural change in the process behavior. We devise an algorithm that detects this departure accurately and efficiently; *(iii)* we create new carefully crafted attacks ranging from easy to detect to fairly stealthy, explore their impact on the TE process, and make the attack data publicly available; and *(iv)* we validate our approach by conducting extensive experiments using a simulation platform, data from a physical testbed, and network traffic from a real ICS. Moreover, we compare PASAD with a popular existing method and show that the results are in favor of our approach.

In §2, we describe PASAD in detail, treating both its theoretical and practical aspects. We establish a framework for validating our approach in §3. Then, we describe and discuss the experiments and the results thereof in §4. In §5, we review related literature, and finally, we conclude this work in §6.

2 PASAD: PROCESS-AWARE STEALTHY ATTACK DETECTION

PASAD is an anomaly-based process-level intrusion detection system that monitors ICS process activity in real time to determine whether the system operation is normal or anomalous. Initially, PASAD learns the normal behavior recorded in a time series of sensor measurements through a training phase, during which ideas from singular spectrum analysis are applied to extract signal information from process output under normal conditions. Thereafter, the system continuously checks if incoming observations are departing from the normal behavior captured during the training phase. The basic idea behind *departure detection* is explained in §2.4.

PASAD consists of four steps formally defined in §2.3. In the first step, the time series of sensor measurements is embedded in a Euclidean space, which means that PASAD mostly deals with vectors and vector spaces. Therefore, after we motivate our approach in §2.1, we proceed by introducing preliminary concepts in linear algebra in §2.2. In the second step, a spectral decomposition of a special matrix derived from the time-series data is performed to extract the deterministic part of the system behavior. In the third step, a signal subspace is identified and vectors corresponding to sensor measurements during normal operating conditions are projected onto this subspace to obtain a representation of the normal process behavior. We provide a mathematical interpretation of projecting vectors onto the signal subspace in §2.5. In the final step, PASAD keeps track of a departure score in order to determine if the process is departing from the normal state. In §2.6, we point out the mathematical property that leads to the *isometry trick* and the two resulting benefits mentioned in §1, which we thoroughly discuss in §2.7 and §2.8. Finally, we discuss the choices for the parameters involved in §2.9 and PASAD's performance in §2.10.

2.1 Motivation

A key enabling property for PASAD is that ICS exhibit **regular dynamics**. They tend to have static topologies and regular communication patterns [11]. A typical control system utilizes sensors, actuators, and controllers to regulate some controlled process. Sensor devices measure some physical property and communicate the measurements to a controller (e.g., a PLC), which based on a control algorithm, correspondingly issues commands to actuators (e.g., control valves) that directly manipulate the physical process based on the received commands. ICS perform clearly defined tasks with clear control objectives and in a well-controlled environment. They typically consist of a number of control loops, each regulating some physical property by trying to maintain its measurements around a predefined set point. A closed-loop control system is *fully automated* and performs almost exclusively without human intervention, and PLCs behave in a cyclic manner [54]. This effectively means that the same dynamics repeat constantly over time. Consequently, even in the presence of noise, the level of *determinism* in the behavior of control systems is relatively high. Sometimes the control loops are nested and cascading, where the set point of one loop is based on the process variable determined by another loop. Even then, the dynamics are likely to repeat, since the loops operate continuously over the duration of the process with *predetermined cycle times* [48].

Mo and Sinopoli [38] point out the key problem of making the unrealistic assumption that the system model is noiseless when developing techniques to identify malicious behaviors in control systems. As physical process control variables may exhibit noisy behaviors by nature [15], they argue that in a noisy environment, strategic attackers have the advantage of inflicting a large perturbation on the system state while avoiding detection by failure detectors and anomaly detectors that do not account for noise. The goal of strategic attackers is to cause slow damaging perturbations in the physical process while remaining unnoticed, so that it runs in a suboptimal setting, eventually leading to performance degradation. The expected outcome of such attacks is to cause a cascading effect due to the interaction between control loops to eventually induce a complete failure of the control system. PASAD is capable of capturing the deterministic behavior of the physical process, despite the fact that the environment in which control systems operate is noisy.

To identify and obtain a mathematical representation of the deterministic behavior, we partly base our technique on singular spectrum analysis—a model-free time-series analysis tool. Vautard and Ghil [52] describe how in nonlinear dynamical systems, the interaction of a large number of degrees of freedom gives rise to what is called "deterministic chaos" in time series. Then, they argue how SSA can tell much about what deterministic part of the system behavior *recorded* in a time series is due to a *few* degrees of freedom and what chaotic part is due to the many rest. They refer to the number of degrees of freedom in the former case as the **statistical dimension** of the time series. In essence, this dimension is determined by solving an eigenvalue problem of a covariance matrix derived from the data. We describe the procedure in detail in §2.3, but first we introduce preliminary concepts in linear algebra.

2.2 Preliminaries & Notation

Definitions and derivations of some of the mathematical concepts that we state without proof throughout this paper can be found in many books on linear algebra, e.g., [49].

A set B of vectors is said to *span* a vector space V if every vector in V is a linear combination of the vectors in B. The set B is *linearly independent* if none of its vectors is a linear combination of the other vectors, and *orthonormal* if its vectors have unit length and are pairwise orthogonal. The set B is said to be a *basis* for V if it

both spans V and is linearly independent; it is an *orthonormal basis* if, in addition, its vectors are orthonormal.

Let $\mathbf{A}$ be an $m \times n$ matrix whose entries are real. Like any matrix, $\mathbf{A}$ is associated with four *fundamental subspaces*: its *range* $\mathcal{R}(\mathbf{A})$ (or column space), its *kernel* $\mathcal{K}(\mathbf{A})$ (or null space), the range of its transpose $\mathcal{R}(\mathbf{A}^T)$ (or row space of $\mathbf{A}$), and the kernel of its transpose $\mathcal{K}(\mathbf{A}^T)$ (or left null space of $\mathbf{A}$). For a subspace W of V, the *orthogonal complement* of W is the set $W^\perp$ of all vectors in V that are orthogonal to every vector in W. We will use the fact that

$$\mathcal{K}(\mathbf{A})^\perp = \mathcal{R}(\mathbf{A}^T). \tag{1}$$

For a matrix $\mathbf{A}$ in $\mathbb{R}^{m \times n}$ with $m > n$, and for a vector $\mathbf{x}$ not necessarily in $\mathcal{R}(\mathbf{A})$, the orthogonal projection of $\mathbf{x}$ onto the range of $\mathbf{A}$ is given by $\mathbf{P}_\mathbf{A}\mathbf{x}$ where

$$\mathbf{P}_\mathbf{A} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \tag{2}$$

is a projection matrix. Moreover, every orthogonal projection $\mathbf{P}$ satisfies the two properties

$$\mathbf{P}^2 = \mathbf{P} \quad \text{(idempotence)} \qquad \mathbf{P}^T = \mathbf{P} \quad \text{(symmetry)}. \tag{3}$$

Notation. In what follows, when we refer to $\mathbf{A}$ as a *linear transformation* (or *linear map*), we mean the matrix representation of some linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ such that for all $\mathbf{x}$ in $\mathbb{R}^n$, $T(\mathbf{x}) = \mathbf{A}\mathbf{x}$. We denote by the matrices $\mathbf{A}^g, \mathbf{A}^+ \in \mathbb{R}^{n \times m}$ the *generalized inverse* and the *Moore-Penrose pseudoinverse* of $\mathbf{A}$ respectively. We use boldface lowercase for vectors, and boldface uppercase for matrices; $||\cdot||$ is the Euclidean 2-norm.

2.3 The Four Steps of PASAD

Consider a univariate real-valued time series of sensor measurements $\mathcal{T} = x_1, x_2, \cdots, x_N, x_{N+1}, \cdots$, PASAD consists of the following four steps.

Step 1: (Embedding)
Let L be an integer, referred to as the *lag* parameter, then an initial subseries of $\mathcal{T}$ of length N is embedded in the L-dimensional Euclidean space $\mathbb{R}^L$ by forming K L-lagged vectors

$$\mathbf{x}_i = (x_i, x_{i+1}, \cdots, x_{i+L-1})^T \tag{4}$$

for all $1 \le i \le K$, where $K = N - L + 1$, and constructing the **trajectory matrix**

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_K \\ x_2 & x_3 & \dots & x_{K+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & \dots & x_N \end{bmatrix} \tag{5}$$

whose columns are the lagged vectors.

Step 2: (Singular Value Decomposition)
To extract noise-reduced signal information describing the deterministic behavior of the control system, the Singular Value Decomposition (SVD) of the trajectory matrix $\mathbf{X}$ is performed to obtain the L eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_L$ of the so-called *lag-covariance* matrix $\mathbf{X}\mathbf{X}^T$. Then, the statistical dimension r of the time series—the number of degrees of freedom that account for the deterministic variability—is determined (see §2.9).

Step 3: (Projection onto the Signal Subspace)
After the signal information has been obtained, in this step, a mathematical representation of the normal process behavior is identified. Let $\mathbf{U}$ be an L-by-r matrix whose columns are the r eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r$ corresponding to the r leading eigenvalues, and let $\mathcal{L}^r$ be the subspace spanned by the column vectors of $\mathbf{U}$. Compute the sample mean of the lagged vectors $\mathbf{x}_i, 1 \le i \le K$, as

$$\mathbf{c} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{x}_i \tag{6}$$

and the centroid of the cluster they form in $\mathcal{L}^r$ as

$$\tilde{\mathbf{c}} = \mathbf{P}\mathbf{c} \tag{7}$$

where $\mathbf{P} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T = \mathbf{U}\mathbf{U}^T$ is a projection matrix (see §2.5).

Step 4: (Distance Tracking)
To detect attack-indicating structural changes in the system behavior, a departure score is computed for every incoming sensor observation. For every test vector $\mathbf{x}_j$ $(j > K)$, compute the squared Euclidean distance from the centroid in $\mathcal{L}^r$ as

$$D_j = ||\tilde{\mathbf{c}} - \mathbf{P}\mathbf{x}_j||^2. \tag{8}$$

Finally, generate an alarm whenever $D_j \ge \theta$ for some threshold θ.

As implied in Eq. (7) and Eq. (8), the lagged vectors are projected onto the signal subspace $\mathcal{L}^r$. The explicit projection at every iteration in the detection phase to compute the departure score is computationally expensive. In §2.6, we show how such a complexity can be avoided using the isometry trick.

PASAD runs in two phases: an offline training phase and an online detection phase. In the training phase (steps 1-3), the time series is embedded in the L-dimensional Euclidean space, and a signal subspace is identified by determining the statistical dimension of the series and the set of r eigenvectors of the lag-covariance matrix. The lagged vectors used for training—the training vectors—are then (implicitly) projected onto the signal subspace, and the centroid $\tilde{\mathbf{c}}$ of the cluster they form is computed. Then, in the detection phase (step 4), PASAD actively checks if lagged vectors $\mathbf{x}_j$ $(j > K)$—the test vectors—are departing from the cluster by tracking the squared Euclidean distance D_j from the centroid.

As the first two steps are known from singular spectrum analysis, for the sake of brevity, we refer the reader to [19] for a good treatment of the SSA theory and methodology. We extend the theory and adapt it to the problem of detecting attacks on ICS in steps 3 and 4, which we thoroughly treat in the remainder of this section.

2.4 Departure Detection: The Basic Idea

We now describe the basic idea behind departure detection. As explained in §2.3, the first embedding step results in K vectors $\mathbf{x}_i$ that lie in the L-dimensional space $\mathbb{R}^L$—*the trajectory space*. Then, the singular value decomposition of the trajectory matrix $\mathbf{X}$ yields an orthonormal set of L eigenvectors. Some $r < L$ of these eigenvectors, associated with the largest eigenvalues, span an r-dimensional linear subspace $\mathcal{L}^r \subset \mathbb{R}^L$, which we refer to as the *signal subspace*. The matrix $\mathbf{U}$, whose columns are the r orthonormal eigenvectors, is then formed so that $\mathbf{P} = \mathbf{U}\mathbf{U}^T$ is the projection matrix that maps the column vectors $\mathbf{x}_i$ of $\mathbf{X}$ to the subspace $\mathcal{L}^r$. The projected training vectors occupy a dense region in $\mathcal{L}^r$ and thereby form a *cluster*

(see Figure 1). The centroid of the cluster is then computed by finding the vector $\tilde{\mathbf{c}}$ in $\mathcal{L}^r$ that minimizes the average squared Euclidean distance from all projected training vectors.

Departure. As the time series of sensor measurements $\mathcal{T}$ continues beyond N, if the mechanism generating its values (the physical process) has not changed, then new projected lagged vectors in $\mathcal{L}^r$ should lie close to the cluster. Therefore, the distance between these vectors and the centroid of the cluster should remain reasonably small. If, on the other hand, the mechanism generating the time series changes due to some outside action (attacks), then the projected test vectors will be forced to lie further away from the cluster and consequently, the distance between these vectors and the centroid of the cluster is expected to increase. A departure is detected if this distance crosses a prescribed threshold.

In §2.8, we validate the claims we have just made; namely that *(i)* training vectors form a cluster in the signal subspace; *(ii)* test vectors fall close to the cluster under normal process operation; and *(iii)* test vectors depart from the cluster when the process is under attack. In particular, we show that the linear subspace $\mathcal{L}^r$ is *isomorphic* to the r-dimensional Euclidean space, allowing us to visualize the structure of the projected lagged vectors in $\mathbb{R}^3$. Next we provide a mathematical interpretation of projecting the lagged vectors onto the signal subspace.

2.5 Projection onto the Signal Subspace

In the process of obtaining a mathematical representation of the normal process behavior, the training vectors are projected onto the signal subspace $\mathcal{L}^r$ (step 3). Here, we give a mathematical interpretation of this projection. In §2.7, we show that this projection is in fact *implicit* as a result of the isometry trick.

The r eigenvectors obtained from the SVD of the trajectory matrix $\mathbf{X}$ form an *orthonormal basis* for the signal subspace (see §2.2), which is presumed to contain most of the signal information recorded in the time series of sensor measurements. The central idea of PASAD is then to check whether or not current sensor observations, in the form of lagged vectors in the trajectory space, conform with the information obtained about the signal during normal process operation, in the form of a subspace of the trajectory space. Naturally, since $\mathcal{L}^r$ is a subspace of $\mathbb{R}^L$ and every lagged vector $\mathbf{x}$ resides in $\mathbb{R}^L$, we opt to find the *best* representation of $\mathbf{x}$ in $\mathcal{L}^r$. By best representation we mean the vector $\mathbf{p}$ in $\mathcal{L}^r$ that is the *closest* possible to $\mathbf{x}$, in the sense that $||\mathbf{p}-\mathbf{x}||$ is minimal. It is well known that this best approximation vector is given by the orthogonal projection $\mathbf{p}=\mathbf{P}\mathbf{x}$, where $\mathbf{P}$ is the projection matrix onto $\mathcal{L}^r$ and $||\mathbf{p}-\mathbf{x}||$ is the projection error.

Since by construction, the signal subspace is the same as the range of the matrix $\mathbf{U}$, the projection matrix that maps vectors in $\mathbb{R}^L$ to $\mathcal{L}^r$ is given by $\mathbf{P}=\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$ (see Eq. (2)). Evidently, for the projection matrix $\mathbf{P}$ to be defined, the columns of $\mathbf{U}$ must be linearly independent so that $\mathbf{U}^T\mathbf{U}$ is invertible. The column vectors $\mathbf{u_i}$ of $\mathbf{U}$, resulting from the SVD of the trajectory matrix $\mathbf{X}$ (step 2), are not only linearly independent, but also orthonormal. It follows that $\mathbf{U}^T\mathbf{U}=\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, and the projection matrix is thus reduced to $\mathbf{P}=\mathbf{U}\mathbf{U}^T$.

As the best representation of the lagged vectors in the signal subspace can be obtained by an orthogonal projection, we first project the training vectors onto $\mathcal{L}^r$ and compute the centroid of the resulting cluster (see Eq. (7)). Then, to detect malicious changes in the process behavior, PASAD checks, in real time, if the most recent sensor observations are persistently departing from the cluster of projected training vectors in the signal subspace (step 4). This is done by keeping track of the distance between the most recent test vector and the centroid of the cluster.

2.6 The Isometry Trick

In a nutshell, the isometry trick states that, for an arbitrary vector $\mathbf{x}$ in $\mathbb{R}^L$, computing the norm of the vector $\mathbf{U}^T\mathbf{x}$ has the effect of *implicitly* projecting $\mathbf{x}$ onto the subspace $\mathcal{L}^r$ and computing its norm there.

To make progress, consider the linear map $\mathbf{U}^T:\mathbb{R}^L\to\mathbb{R}^r$ and note that for all $\mathbf{x}$ in $\mathbb{R}^L$, the following equality holds

$$||\mathbf{U}^T\mathbf{P}\mathbf{x}||=||\mathbf{P}\mathbf{x}|| \tag{9}$$

(see Appendix A for a proof). Informally, Eq. (9) implies that whenever a vector $\mathbf{x}$ is projected onto $\mathcal{L}^r$, the resultant vector is one whose length does not change when acted upon by the transformation $\mathbf{U}^T$. Formally, first note the key property

$$\mathcal{R}(\mathbf{P})=\mathcal{R}(\mathbf{U}) \tag{10}$$

(see Appendix B for a proof) and that $\mathcal{K}(\mathbf{U}^T)^\perp=\mathcal{R}(\mathbf{U})$ (from Eq. (1)), then $\mathbf{U}^T$ is said to be a **partial isometry** [24].

Definition (Partial Isometry). *A not necessarily square matrix* $\mathbf{A}$ *is called a partial isometry if the vectors* $\mathbf{v}$ *and* $\mathbf{A}\mathbf{v}$ *have the same Euclidean norm whenever* $\mathbf{v}$ *is in the orthogonal complement of the kernel of* $\mathbf{A}$.

A (linear) isometry between two normed vector spaces is a linear map that *preserves* the length of vectors (and by linearity, the distance between two vectors) for all vectors in its domain. Note that $\mathbf{U}:\mathbb{R}^r\to\mathbb{R}^L$ is an isometry, since for every $\mathbf{y}\in\mathbb{R}^r$, $||\mathbf{U}\mathbf{y}||=||\mathbf{y}||$.[1] On the other hand, $||\mathbf{U}^T\mathbf{x}||$ and $||\mathbf{x}||$ are not equal in general. However, when the domain of $\mathbf{U}^T$, namely $\mathbb{R}^L$, is restricted to the orthogonal complement of its kernel, $\mathbf{U}^T$ becomes an isometry as asserted by Eq. (9). Indeed, for every $\mathbf{x}$ in $\mathbb{R}^L$, let $\mathbf{v}=\mathbf{P}\mathbf{x}$, then $||\mathbf{U}^T\mathbf{v}||=||\mathbf{v}||$, for every $\mathbf{v}\in\mathcal{R}(\mathbf{P})=\mathcal{R}(\mathbf{U})=\mathcal{K}(\mathbf{U}^T)^\perp$.

But $(\mathbf{U}^T)^g=\mathbf{U}$; that is, $\mathbf{U}$ is a generalized inverse of $\mathbf{U}^T$, since

$$\mathbf{U}^T=\mathbf{U}^T\mathbf{U}\mathbf{U}^T=\mathbf{U}^T\mathbf{P} \tag{11}$$

which follows directly from the fact that $\mathbf{U}^T\mathbf{U}=\mathbf{I}$. Hence, by virtue of Eq. (11), one can rewrite Eq. (9) as

$$||\mathbf{U}^T\mathbf{x}||=||\mathbf{U}\mathbf{U}^T\mathbf{x}||, \tag{12}$$

suggesting that computing norms in $\mathcal{L}^r$ can be done without the need for an explicit projection.

We refer to using the property that $\mathbf{U}^T$ is a partial isometry to compute the norms $||\mathbf{U}^T\mathbf{x}||$ in $\mathbb{R}^r$ instead of the norms $||\mathbf{U}\mathbf{U}^T\mathbf{x}||$ in $\mathcal{L}^r$ as the *isometry trick*.

Next, we show that this property allows for more efficient tracking of the distance between the test vector and the centroid during the detection phase. In §2.8, we extend this result to show that $\mathcal{L}^r$

[1] Since $||\mathbf{U}\mathbf{y}||^2=(\mathbf{U}\mathbf{y})\cdot(\mathbf{U}\mathbf{y})=\mathbf{y}^T\mathbf{U}^T\mathbf{U}\mathbf{y}=\mathbf{y}^T\mathbf{y}=||\mathbf{y}||^2$, $\forall\,\mathbf{y}\in\mathbb{R}^r$.

is isomorphic to the r-dimensional Euclidean space. A key benefit of this isomorphism is the ability to visualize the normal behavior as a cluster of lagged vectors in the signal subspace, as well as the departure from the cluster when the process is under attack.

2.7 Efficiency: Implicit Projection

In the detection phase, a departure score for the most recent test vector is obtained by iteratively computing the distance D_j from the centroid determined in the training phase (see Eq. (8)) .

By writing $D_j = ||\mathbf{P}(\mathbf{c}-\mathbf{x}_j)||^2 = ||\mathbf{U}\mathbf{U}^T(\mathbf{c}-\mathbf{x}_j)||^2$, we observe that computing the distance from the centroid of the cluster amounts to computing the (squared) norm of the projection of the difference vector $(\mathbf{c}-\mathbf{x}_j)$ onto $\mathcal{L}^r$. From the isometry trick, we learned that in order to compute the norm of the projection of any vector $\mathbf{x}$ in the trajectory space, it is sufficient to compute the norm of $\mathbf{U}^T\mathbf{x}$ without the need for an explicit projection onto $\mathcal{L}^r$. With this in mind, computing $\mathbf{U}^T\mathbf{x}$ instead of $\mathbf{U}\mathbf{U}^T\mathbf{x}$ at every iteration leads to a significant gain in performance (see §2.10 for more performance analysis), which increases the deployability of PASAD on limited-resource hardware, and its applicability to real industrial settings.

Thus, we can now efficiently compute the centroid as $\tilde{\mathbf{c}} = \mathbf{U}^T\mathbf{c}$ and the departure score as $D_j = ||\tilde{\mathbf{c}} - \mathbf{U}^T\mathbf{x}_j||^2$, replacing Eq. (7) and Eq. (8) respectively in the actual implementation. Consequently, the departure score can be evaluated more efficiently in the low-dimensional space $\mathbb{R}^r$, while still gaining the luxury of computing the distance from the cluster in the signal subspace.

2.8 Validation: Visualizing the Departure

We now extend the result in Eq. (12) to show that by restricting the domain of $\mathbf{U}^T$ to $\mathcal{R}(\mathbf{U}) = \mathcal{R}(\mathbf{P})$, we obtain an isomorphism between the signal subspace and a low-dimensional Euclidean space, which allows us to visualize the behavior of the underlying process. We do not make the claim that the ability to visualize the departure of the process in ICS would offer plant engineers a complete picture of the causes and physical implications of the attack. However, we consider that the chief advantage of being able to visualize the time-series data and the departure is that it empirically *validates* the theoretical claims about PASAD that we have made in §2.4.

We showed in §2.6 that the linear map $\mathbf{U}^T$, being a partial isometry, preserves the vector norm when its domain is restricted to $\mathcal{R}(\mathbf{U})$. Let $\tilde{\mathbf{U}} : \mathcal{R}(\mathbf{U}) \to \mathcal{R}(\mathbf{U}^T)$ be the restricted map, i.e., $\tilde{\mathbf{U}} = \mathbf{U}^T\big|_{\mathcal{R}(\mathbf{U})}$, then evidently $\tilde{\mathbf{U}}$ is an isometry. With the following proposition, we show that $\tilde{\mathbf{U}}$ is further an isomorphism between the vector spaces $\mathcal{L}^r$ and $\mathbb{R}^r$. The signal subspace being isomorphic to the r-dimensional Euclidean space effectively means that the two vector spaces are fundamentally the "same" for all practical purposes (e.g., with respect to dimension, linear independence, vector norm, distance between vectors, linear combinations, etc). The key benefit for PASAD then is that it can operate in a simpler space where computing the distance from the centroid can be done more efficiently and where the time-series data can be visualized.

PROPOSITION. *The restricted linear map* $\tilde{\mathbf{U}} : \mathcal{R}(\mathbf{U}) \to \mathcal{R}(\mathbf{U}^T)$ *is an isometric isomorphism and admits a Moore-Penrose pseudoinverse* $\tilde{\mathbf{U}}^+ = \mathbf{U}$. $\mathcal{R}(\mathbf{U})$ *and* $\mathcal{R}(\mathbf{U}^T)$ *are therefore isometrically isomorphic* ($\mathcal{R}(\mathbf{U}) \cong \mathcal{R}(\mathbf{U}^T)$).

For a sketch of the proof (see Appendix C for a complete proof), note that a linear isometry between two normed vector spaces is a (linear) isomorphism if it is bijective (i.e., both injective and surjective). $\tilde{\mathbf{U}}$ is injective since it is left-invertible as $\tilde{\mathbf{U}}^+\tilde{\mathbf{U}} = \mathbf{U}\mathbf{U}^T = \mathbf{P}$, where $\mathbf{P}$ is the identity map on $\mathcal{R}(\mathbf{U})$ (by the idempotence property in (3)). Moreover, $\tilde{\mathbf{U}}$ is surjective since it is right-invertible as $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^+ = \mathbf{U}^T\mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity map on $\mathcal{R}(\mathbf{U}^T)$. Hence, $\tilde{\mathbf{U}}$ is a bijective isometry, and thus an isomorphism.

Finally, since all r columns of $\mathbf{U}$ are linearly independent, $\mathbf{U}$ has *full rank*, meaning that its kernel contains only the zero vector. Therefore, by Eq. (1), $\mathcal{R}(\mathbf{U}^T) = \mathbb{R}^r$, and as $\mathcal{R}(\mathbf{U}) = \mathcal{L}^r$ by construction, it follows, by the previous proposition, that $\tilde{\mathbf{U}} : \mathcal{L}^r \to \mathbb{R}^r$ is an isomorphism and consequently

$$\mathcal{L}^r \cong \mathbb{R}^r. \tag{13}$$

While vectors in the r-dimensional signal subspace $\mathcal{L}^r$ have L components (since $\mathcal{L}^r \subset \mathbb{R}^L$), vectors in the range of the partial isometry $\mathbf{U}^T$ have only r components (since $\mathcal{R}(\mathbf{U}^T) = \mathbb{R}^r$). Thus, by Eq. (13), images of all vectors in the trajectory space can be expressed with respect to the standard basis for $\mathbb{R}^r$. By choosing only the first 3 basis vectors of the signal subspace, empirically deemed sufficient for capturing the main structure, we can plot the data vectors as points in $\mathbb{R}^3$ and visualize the structure.

Figure 1 depicts a visualization of the departure of the Tennessee-Eastman process from normal operating conditions when the process is under a stealthy type-$\mathcal{SA}$**3** attack (described in detail in §3.1.1). The upper-left subplot shows a time series of raw measurements corresponding to the sensor being monitored by PASAD. The initial subseries was used for training to extract the basis vectors of the signal subspace, map the training vectors to $\mathbb{R}^3$, and compute the centroid of the cluster they form. As shown on the right, when the process is running under normal operating conditions, the projected test vectors lie close to the cluster, whereas when the process is under attack, the vectors start departing from the cluster. The lower-left subplot shows the values for the distance D_j from the centroid $\tilde{\mathbf{c}}$ which PASAD iteratively computes for every test vector $\mathbf{x}_j$, and how the departure was detected shortly after the attack started.

2.9 Choice of Parameters

There are three parameters involved in PASAD: the length N of the initial part of the time series used for training, the lag parameter L, and the statistical dimension r. For the length of the training subseries, the best practice is to have it sufficiently large so that it incorporates an essential part of the signal in the noisy time series. To determine the lag parameter L and the statistical dimension r, there are standard SSA recommendations and guidelines to follow [19, 39]. For the lag parameter, it should be that $L \leq N/2$, and in practice, the choice $\lfloor L = N/2 \rfloor$ often proves adequate. As for the statistical dimension, we choose r such that the r leading eigenvectors provide a good description of the signal and the lower $L - r$ eigenvectors correspond to noise. The choice of r can be assisted by a *scree plot*,[2] in which the spectrum of the eigenvalues obtained

[2] A *scree plot* is a statistical test, frequently used in *factor analysis*, to estimate the number of factors (eigenvectors in our case) that correspond to most of the variability in the data.

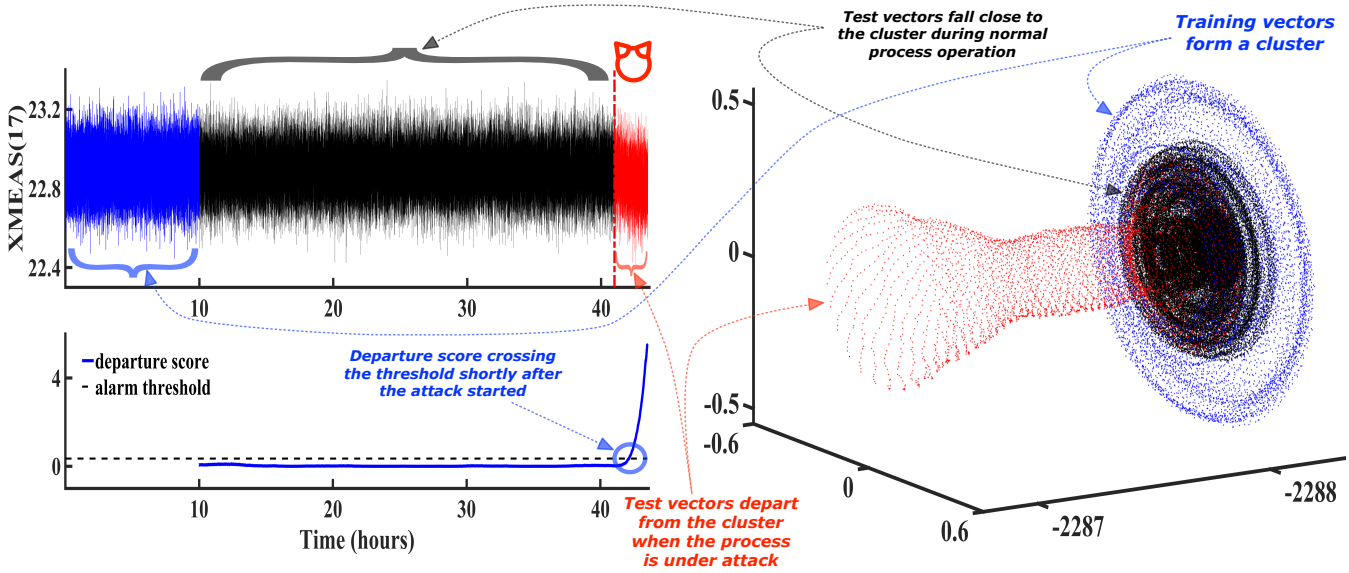

Figure 1: A visualization of the departure of the TE process from the normal state. The upper-left subplot shows the raw sensor measurements. As shown on the right, when the process is running under normal conditions, the test vectors lie close the cluster, whereas when the process is under attack, the vectors start departing from the cluster. The lower-left subplot shows the values for the departure score and how the departure was detected shortly after the attack onset.

in the SVD step reveals the noise level as a flat tail, such that the statistical dimension is the number of eigenvalues above this level.

Finally, to determine the threshold θ, we run PASAD on a *validation series*, which is the subseries observed under normal conditions and preceded by the training subseries, i.e., the subseries determined by the interval $(N+1, \tau-1)$, where τ is the starting time of the attack (when applicable). More formally, we define the threshold as $\theta = M_{n,r,\tau} + \epsilon$ such that $M_{n,r,\tau} = \max_n \{D_{n,r,\tau} : N < n < \tau\}$ where $D_{n,r,\tau}$ is the departure score corresponding to observations in the specified range and ϵ is a relatively small constant. We motivate this approach to determining the alarm threshold in §4.5.

2.10 Implementation & Performance

A pseudocode of PASAD is shown in Algorithm 1. The most expensive computational step during training is computing the singular value decomposition of the trajectory matrix $\mathbf{X}$.[3] In general, the SVD of an $m \times n$ matrix is computable in time proportional to $O(\min\{mn^2, m^2n\})$. In our particular case, $\mathbf{X}$ is of dimension $L \times K$, where $K = N - L + 1$. Then, we have $L \leq N/2$ as mentioned in §2.9, which implies that $L \leq (K+L-1)/2 \Rightarrow L \leq K-1 \Rightarrow L < K$, so that $\min\{LK^2, L^2K\} = L^2K$. Thus the time complexity of the SVD step in PASAD is $O(L^2K)$.

Although the time complexity of the training procedure is quadratic in the size of the lag parameter L, performance is not a critical issue for the training phase since it is an offline procedure. On the other hand, it is crucial that testing on incoming observations can be done efficiently to allow for real-time protection. As mentioned in §2.7, the departure score for the j^{th} test vector is computed as $||\tilde{\mathbf{c}} - \mathbf{U}^T\mathbf{x}_j||^2$. First, $\tilde{\mathbf{c}} - \mathbf{U}^T\mathbf{x}_j$ is evaluated in $O(rL)$, since it involves a product of matrices with dimensions $r \times L$ and $L \times 1$ respectively, then the elements of the resultant vector are squared and added in $O(r)$. However, r is a constant that does not depend on L and typically $r << L$; therefore, the overall time complexity of the detection phase is linear in L.

[3] Note that the HANKEL function on Line 19 is used because the trajectory matrix has a Hankel structure.

3 A FRAMEWORK FOR VALIDATION

In this section, we describe three different scenarios for validating PASAD: the Tennessee-Eastman process, a dataset from the SWaT testbed, and a network traffic from a real water distribution plant.

3.1 Scenario I: The Tennessee-Eastman Process

We have developed a set of new attacks on the Tennessee-Eastman process control model [12] that aim to cause tangible impact on the underlying physical process.[4] To provide an intelligible explanation of the attacks we have designed, a high-level description of the process is in order.

The TE simulation model simulates a real plant-wide chemical process. The process was originally released with no embedded control strategy as the aim of its release was to challenge the control theory community to develop and benchmark different optimized control strategies. Indeed, several strategies have been proposed in response to the challenge [32, 45]. More recently however, acting as both a realistic and safe environment for experimentation, the TE process has transcended its original objective and has come to be a popular choice amongst ICS security researchers [8, 28, 29, 31, 37].

We use the popular simulation model proposed by Downs and Vogel [12], who modified some aspects of the chemical process,

[4] The TE attack data and PASAD's code are made publicly available at https://github.com/mikeliturbe/pasad

Algorithm 1 PASAD: An algorithm for detecting stealthy attacks on control systems.

Required: Training subseries $\mathcal{T}_{train}$ and the lag parameter L.
Outcome: An alarm is raised whenever the departure score crosses a prespecified threshold.

$N \leftarrow$ LENGTH($\mathcal{T}_{train}$)
$\mathbf{U}^T \leftarrow$ PASAD_TRAIN($\mathcal{T}_{train}, N, L$) ▷ The partial isometry
$K \leftarrow N - L + 1$
for $i \leftarrow 1, K$ **do** ▷ Compute the centroid
  $\mathbf{x}_i \leftarrow i^{th}$ *training_vector*
  $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{x}_i$
end for
$\mathbf{c} \leftarrow \mathbf{s}/K$
$\tilde{\mathbf{c}} \leftarrow \mathbf{U}^T\mathbf{c}$
Determine θ ▷ Alarm threshold
for $\mathbf{x}_j \leftarrow$ *current_test_vector* **do**
  $\mathbf{y} \leftarrow \tilde{\mathbf{c}} - \mathbf{U}^T\mathbf{x}_j$
  $D_j \leftarrow \mathbf{y}^T\mathbf{y}$ ▷ The departure score
  if $D_j \geq \theta$ **then** ▷ Check for departure
    generate alarm
  end if
end for

function PASAD_TRAIN($\mathcal{T}_{train}, N, L$)
  $\mathbf{X} \leftarrow$ HANKEL($\mathcal{T}_{train}, N, L$) ▷ Trajectory matrix
  $z \leftarrow$ SVD($\mathbf{X}$) ▷ Solve the SVD problem
  Determine r ▷ The statistical dimension
  $\mathbf{U} \leftarrow z$.eigenvectors(r) ▷ The r leading eigenvectors
  return $\mathbf{U}^T$
end function

such as kinetics, components, and operating conditions, in order to protect its proprietary nature.

The TE process produces two liquid products (G, H) from four gaseous reactants (A, C, D, E), in addition to a byproduct (F) and an inert (B), making a total of eight chemical components, coded after the first eight letters of the alphabet. There are five main operation units: reactor, condenser, recycle compressor, vapor-liquid separator, and stripping column. The gaseous reactants, fed by three different feeds, react to form liquid products. These products, along with residual reactants, leave the reactor as vapors, which are then cooled by the condenser to return to the liquid state. Next, the vapor-liquid separator isolates the non-condensed vapors, which are fed once again to the reactor by using a centrifugal compressor. The condensed components, on the other hand, move to a stripping column to remove the remaining residual reactants. The final product (mix of G and H) exits the stripper and heads towards a refining section that separates its components. This refining section is not included in the model. Similarly, the inert and the byproduct are purged in the vapor-liquid separator as vapor.

The process has 41 measured variables, labeled as XMEAS, that comprise the readings of the sensors, and 12 manipulated variables, named XMV, that correspond to actuator commands. The controller reads the XMEAS values and, based on the implemented control strategy, sends commands to the XMV variables, which mostly represent valves that control different process flows. A full description of all process variables can be found in the original TE paper [12].

We use the open-source DVCP-TE implementation of the TE process,[5] first presented by Krotofil and Larsen [30], which is oriented towards security research and features support for performing attacks on both sensor and actuator signals. DVCP-TE is a Simulink model, where the process is implemented as an S-function. A set of control strategies, along with a framework for performing attacks, are implemented in Simulink blocks.

As stated previously, the TE process originally lacked any control strategy. If the process runs in absence of a control algorithm, it remains unstable and eventually comes to a complete stop due to a too high separator liquid level. In order to keep the process running and maintain its stability, we use the control strategy presented by Larsson et al. [32].

Most of the previous work focuses on the unreliability of the sensor readings, while trusting actuators and controllers [51]. By contrast, we simulate *integrity attacks* on both sensors and actuators as depicted in Figure 2. Once attackers gain access to a control network in charge of a process, they can either compromise the data fed to the controller by tampering with the process readings transmitted by the sensors, or tamper with the commands sent by the controller to the actuators. In the former case, the controller makes decisions based on maliciously modified data, potentially leading to the destabilization of the process. In the latter case, the process acts on arbitrary commands sent by the attacker rather than on the commands sent by the controller.

We use the attacker model proposed by Krotofil et al. [31], where we simulate measured variables $u_i'(t)$ and $y_i'(t)$ as

$$u_i'(t), y_i'(t) = \begin{cases} u_i(t), y_i(t) & \text{for } t \notin T_a \\ u_i^a(t), y_i^a(t) & \text{for } t \in T_a \end{cases} \quad (14)$$

such that $u_i(t), y_i(t)$ and $u_i^a(t), y_i^a(t)$ correspond to the i^{th} original

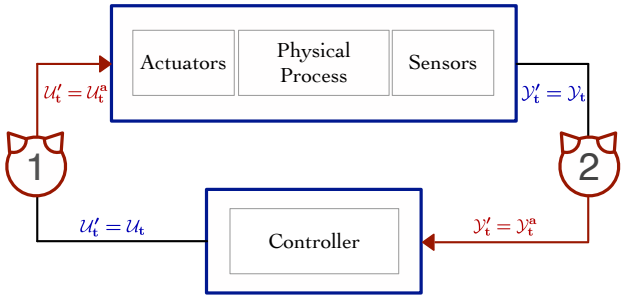

Figure 2: Attack scenarios on control systems: Attacks on actuator signals (1) and attacks on sensor signals (2).

and modified measured variables at time $0 \leq t \leq T$ respectively, T is the duration of the simulation run, and T_a is the attack interval.

We have designed the attacks with two main objectives in mind: *(i) Stealth attacks*, designed to cause slow damaging perturbations and aim to degrade the performance of the process; and *(ii) direct damage attacks* where the attacker's goal is to cause damage to physical equipment (e.g., reactor, stripping column, pipes, etc.) that

[5] Available at https://github.com/satejnik/DVCP-TE

is essential for the process to run, mainly by driving the process to unsafe operating conditions (e.g., high temperature or pressure).

3.1.1 Stealth Attacks. In stealth attacks, attackers try to remain undetected by keeping the process readings under a set of thresholds, which if exceeded, alarms are raised and operators are alerted. In the following, we describe three such attacks.

$\mathcal{SA}$1: The manipulated variable XMV(9) corresponds to the valve that controls the steam input to the stripping column. When the TE process is controlled by the control strategy of Larsson et al. [32], no steam is used in the process, and this valve is always closed. In this attack, we consider that the attacker opens the steam valve at 40%. Compared to a completely open steam valve, this attack has less impact on the plant operation, but is nonetheless stealthier.

$\mathcal{SA}$2: The manipulated variable XMV(6) corresponds to the purge valve that controls the output of accumulating reactor gases. Opening this valve more than necessary would result in products being wasted, since in order to maintain the production rate, more reactants would need to be purged from the reactor and fed to the process. However, opening the purge valve too much would drive the reactor pressure to a too low level, causing the process to halt. In this scenario, we set XMV(6) to 28% open, which is wide enough to degrade the performance of the process without interrupting the process execution.

$\mathcal{SA}$3: In this sensor attack, we tamper with the readings of the XMEAS(10) sensor, which measures the purge rate, so that it constantly sends the value zero, tricking the controller into thinking that there is no purge. The controller would then open the valve XMV(6) to counteract.

3.1.2 Direct Damage Attacks. Direct damage attacks aim to sabotage equipment and eventually lead to the interruption of the process. In the following, we describe two such attacks.

$\mathcal{DA}$1: The manipulated variable XMV(10) corresponds to the valve controlling the cooling water flow to the reactor to prevent its pressure from reaching dangerous levels. Therefore, it is a critical valve in the process. In this scenario, we set the XMV(10) valve to 35.9% open, slightly less than the optimal setting. Consequently, the pressure adds up inside the reactor and the TE process execution eventually stops due to reaching the predefined safety limits.

$\mathcal{DA}$2: In this scenario, the value of the sensor XMEAS(7) measuring the reactor pressure is set to zero, so that the controller thinks the pressure is significantly lower than it actually is. In response, the controller opens the way for more reactants to the reactor, thus accelerating the chemical reaction, and eventually increasing the reactor pressure. As in $\mathcal{DA}$**1**, the model stops execution after reaching a too high reactor pressure.

3.2 Scenario II: The SWaT Dataset

The SWaT dataset [16] is a collection of process readings and network traces from the Secure Water Treatment (SWaT) testbed.[6] Constructed to support ICS security research, the SWaT dataset is divided into two main parts: a seven-day-long capture under normal operating conditions and a four-day-long capture while conducting diverse attacks.

The SWaT testbed [36], which was used to build the dataset, is a real scaled-down version of a waste-water treatment plant. The treatment process consists of six different treatment phases, each independently controlled by a PLC, and has the capacity of filtering 18.93 water litres per hour. As such, the testbed is able to faithfully recreate the operation of a real waste-water treatment plant, albeit at a manageable scale.

At the process level, the captured data corresponds to 51 sensor and actuator signals. During the capture, the SWaT testbed undergoes 41 different attacks, five of them with no physical impact. A detailed description of the attacks and their impact on the process, as well as more technical information on the testbed, can be found in [5, 16, 36].

3.3 Scenario III: A Water-Distribution Plant

In the final scenario, the data consists of network traffic captured from equipment inside an operational water distribution plant in Sweden. The 105GB Modbus/TCP traffic capture was recorded over a period of 106 days.

To emulate a real-world scenario, we have set up a testbed consisting of two single-board computers, one switch with a port-mirroring feature, and a router. The ICS network traffic was replayed from network dump files. Then, to capture and parse the traffic, we have built a packet-capturing subsystem on top of Bro [43] that features a dynamic buffering mechanism to deliver process data to PASAD, after having parsed the packets and extracted the register data. Aside from replaying the traffic, all other subsystems, including a C implementation of the detection component of PASAD, were running on a prototype consisting of a single-board computer.

Furthermore, in order to investigate the deployability of PASAD in real environments, we have deployed and run the prototype in an operational paper factory and analyzed its performance [4].

4 EXPERIMENTS & RESULTS

In this section, we first investigate the *time to detection* and the *detection accuracy* of PASAD (EXP. I-III). Then, we compare PASAD with the AR-based method (EXP. IV) to highlight its distinctive capability of detecting stealthy attacks. In EXP. V, we demonstrate the ability of PASAD to maintain a relatively low *false alarm rate* by determining the alarm threshold as defined in §2.9. Finally, in EXP. VI, we explore the applicability of PASAD to real-world scenarios.

In all subsequent figures, the upper subplot shows the raw sensor readings and the lower subplot displays the departure scores. The initial part of the time series that was used for training corresponds to sensor measurements collected while the process is running normally (busy, not idle) and is highlighted in blue, and the extracted signal is highlighted in green.[7] The dotted horizontal line corresponds to the threshold level, and the shaded region corresponds to the time interval (T_a) during which the attack was occurring. The vertical line indicates when the attack started. The selection of

[6] The dataset was collected by the iTrust group from the Centre for Research in Cyber Security in the Singapore University of Technology and Design, and is available upon request at https://itrust.sutd.edu.sg/research/dataset/

[7] To produce the extracted signal, the training vectors are first projected onto the signal subspace, then an approximated series is reconstructed by applying the *diagonal averaging* step in SSA [19].

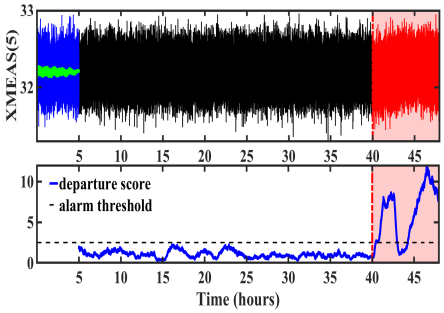

Figure 3: Stealthy attack $\mathcal{SA}1$ compromising the control variable XMV(9) detected in sensor variable XMEAS(5).

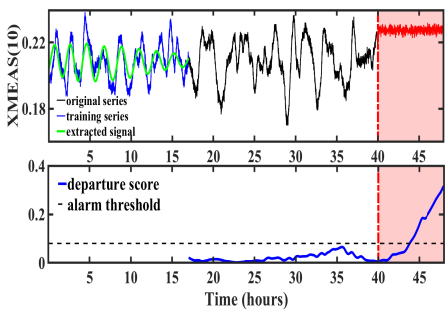

Figure 4: Stealthy attack $\mathcal{SA}2$ compromising the control variable XMV(6) detected in sensor variable XMEAS(10).

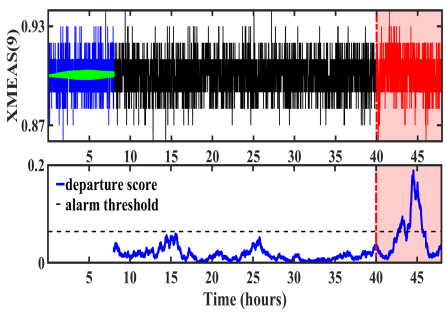

Figure 5: Stealthy attack $\mathcal{SA}3$ compromising the sensor variable XMEAS(10) detected in sensor variable XMEAS(9).

PASAD's free parameters for all the experiments described herein were chosen according to §2.9, and can be found in Table 1.

4.1 EXP. I: Detection of Stealth Attacks

In EXP. I, we wish to detect the three stealth attacks $\mathcal{SA}1$, $\mathcal{SA}2$, and $\mathcal{SA}3$ defined in §3.1.1, which were designed to mimic an adversary whose aim is to cause perturbations in the physical process while remaining unnoticed. This can be achieved by strategic attackers who try to minimize the changes in the time series, optimally hiding the entire change within the noise level, whilst accumulating an impact on the infrastructure by drifting the process from the optimal setting. Figures 3, 4, and 5 depict the detection of stealthy attacks compromising the manipulated variables XMV(9) and XMV(6), and the sensor variable XMEAS(10) respectively.

Stealthy attacks are detectable by PASAD. When the process is under a stealthy attack, the values of the sensor readings remain roughly within the normal range. The process variables, however, do exhibit changes in the oscillation and trend structure. These structural changes would hinder the optimal execution of the control process, leading to performance degradation. As the figures show, PASAD is effective against different test scenarios. Notably, in Figures 3 and 5, it is apparent that covert attacks like $\mathcal{SA}1$ and $\mathcal{SA}3$ exhibit no visual change in the sensor readings but are nevertheless detected by PASAD with a decent time to detection. The drop of the departure score below the threshold in Figure 3 can be explained by the reaction of the control algorithm to the attack, which tries to recover the optimal state of the control process.

4.2 EXP. II: Detection of Direct Damage Attacks

In EXP. II, we apply PASAD to the direct damage attack scenarios $\mathcal{DA}1$ and $\mathcal{DA}2$ described in §3.1.2. These attacks aim to drive the pressure in the reactor to dangerous levels in an attempt to cause irreparable physical damage to the control system. The goal of this experiment is to detect these attacks before such damage occurs. In both attack scenarios, the process reaches its safety limits roughly 8 hours after the attack onset. The detection results of direct damage attacks compromising the manipulated variables XMV(10) and the sensor variable XMEAS(7) are displayed in Figures 6 and 7.

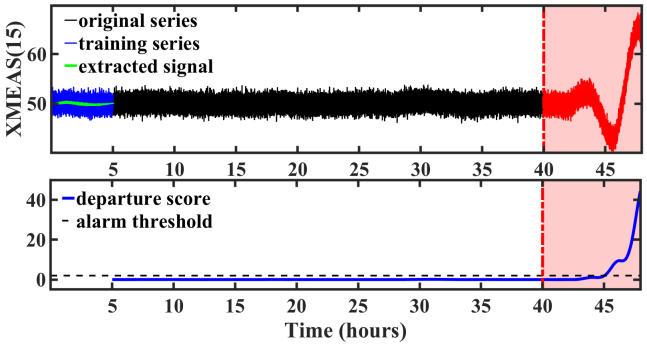

Figure 6: Direct damage attack $\mathcal{DA}1$ compromising the control variable XMV(10) detected in sensor variable XMEAS(15).

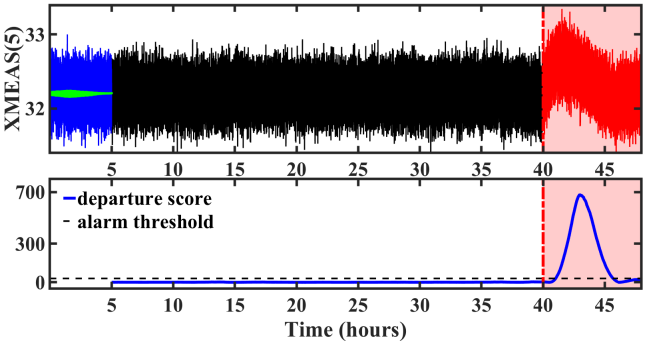

Figure 7: Direct damage attack $\mathcal{DA}1$ compromising the sensor variable XMEAS(7) detected in sensor variable XMEAS(5).

Direct damage attacks are trivial to detect. As Figures 6 and 7 reveal, the impact of direct damage attacks $\mathcal{DA}1$ and $\mathcal{DA}2$ on the process is obvious. Here, the attackers are not trying to remain unnoticed; rather their goal is to cause as much physical damage as they can in as little time as possible by driving the process to an unsafe state. The changes in the behavior of the process variables caused by these attacks were trivially detected by PASAD. The departure score reached the threshold level long before the process could reach the safety limits. The reason why it is fairly trivial to detect this kind of attacks using PASAD, as well as existing methods that monitor residuals, is that they tend to cause significant trends and mean shifts in an otherwise trendless time series. Note that

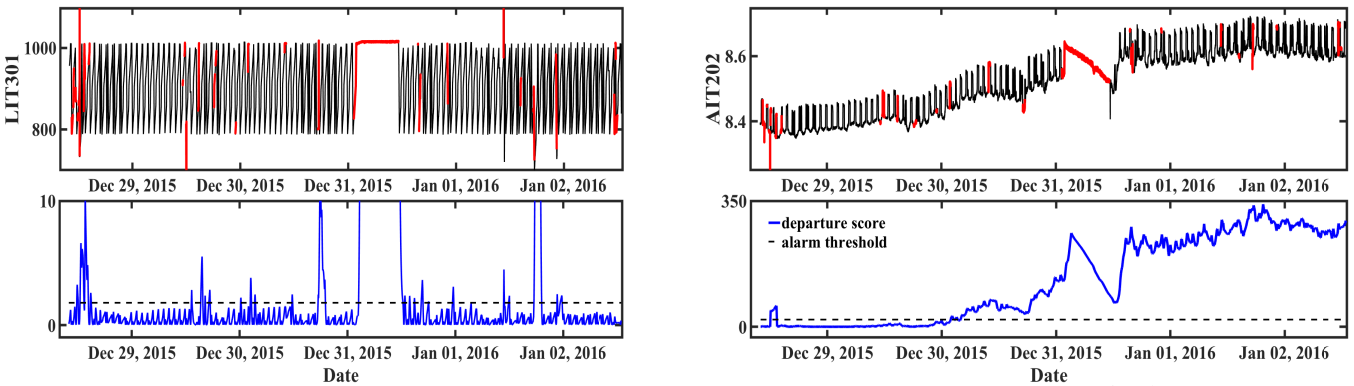

Figure 8: Detection of various attacks on the ultra-filtration feed water tank level sensor signal LIT301 (left) and the water pH analyzer sensor signal AIT202 (right) performed on the SWaT testbed.

after the $\mathcal{DA}2$ attack has been detected, the departure score drops at the same time that the sensor recovers the original behavior.

4.3 EXP. III: Detection of SWaT Attacks

In EXP. III, we run PASAD on the SWaT dataset captured from a scaled-down version of a waste-water treatment plant described in §3.2. Figure 8 shows the evolution of two sensor signals, the ultra-filtration feed water tank level (LIT301) and the water pH analyzer (AIT202), and the corresponding departure scores computed by PASAD. The time intervals during which the testbed is being actively attacked are highlighted in red.

PASAD is process-agnostic. The fact that PASAD performs well on data collected by a third party, where we had little to no insight into the underlying process and the performed attacks, ascertains the validity of our claim about the applicability of our method to a wide range of processes. As shown in Figure 8, PASAD is able to detect attacks on ICS that were performed in a realistic setting and had actual impact on a physical process. Initially, training was performed on separately provided sensor readings that were declared normal. The measurements used for testing were generated while the testbed was subject to a series of attacks, the great majority of which were detected by PASAD including, curiously, attacks that caused no apparent change in the behavior of the LIT301 sensor (left). Note that the AIT202 signal (right) slowly drifts away from its normal range after the attack onset and does not return to its desired setpoint around 8.4 and, correspondingly, the departure score remains consistently above the threshold.

4.4 EXP. IV: Comparison with the AR Method

We further evaluate PASAD by comparing it with the popular Auto-Regression method. An AR model of order p states that at a given time k, the sensor value y_k may be predicted by computing $\hat{y}_k$ as a linear combination of the previous p values plus an error term [6]. As recommended in [51], we extend the *stateless* AR statistic r_k into a *stateful* one by computing an additional CUmulative SUM (CUSUM) statistic, defined as $S_0 = 0$, $S_{k+1} = (S_k + |r_k| - \delta)^+$, where $(x)^+$ means $\max(0, x)$ and δ is a constant chosen such that the expected value of $|r_k| - \delta < 0$ under normal conditions. To estimate

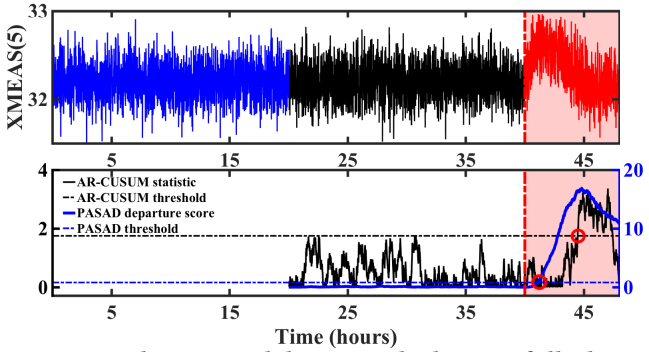

Figure 9: Both PASAD and the AR method successfully detect the direct damage attack $\mathcal{DA}2$.

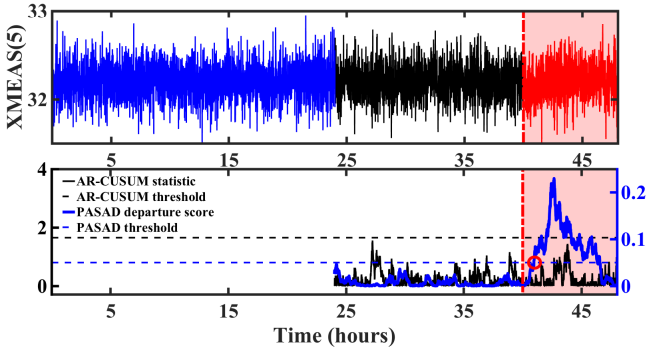

Figure 10: Unlike PASAD, the AR method fails to detect the stealth attack $\mathcal{SA}2$.

the coefficients and the order of the model, we leverage the open-source implementation used in [22] to detect stream deviations in the process variables of water treatment plants.[8] The outcome of the comparison is presented in Figures 9 and 10.

PASAD outperforms AR in both attack scenarios. The comparison conducted in EXP. IV indicates that PASAD outperforms the AR-based technique. Both methods were applied on the same time series under the same attack scenario, and the same subseries was

[8] Available at: https://github.com/RhysU/ar

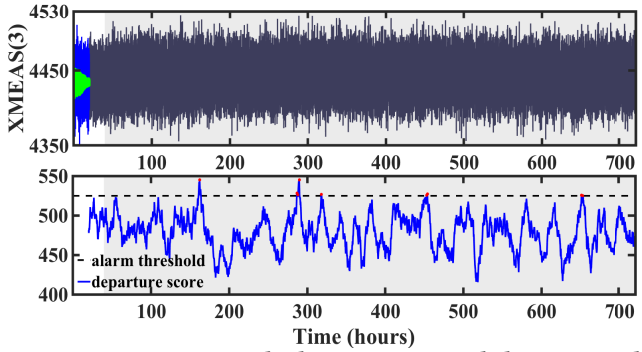

Figure 11: PASAD is applied on a one-month long normal data and only a few false alarms were observed.

used for both training PASAD and inferring the AR model order and coefficients. In Figure 9, it is evident that although both methods are able to successfully detect the direct damage attack, PASAD's departure score behaves statistically better in detecting the attack than the AR-based CUSUM statistic. More specifically, PASAD exhibits a more stable behavior in terms of low variance under normal conditions, and then reacts more promptly to the changes in the time series. More notably, Figure 10 succinctly shows how unlike PASAD, the AR method fails to detect the stealthy attack.

4.5 EXP. V: Validating the Choice of Threshold

Given that the TE process takes quite some time to reach an unsafe state [29], we choose a threshold level as defined in §2.9, so that an attack may be detected in reasonable time, while not raising any false alarms. In EXP. V, PASAD is applied on attack-free sensor measurements from the TE process (Figure 11). The first 20 hours were used for training and estimating the signal subspace. Afterwards, the threshold was chosen to be just above the maximum value attained during the first 40 hours, and the behavior of the departure score was examined over the remaining validation period.

PASAD maintains a low false positive rate. The purpose of choosing the threshold level as defined in §2.9 is to minimize the number of false alarms, which if intolerably high, may hinder the deployment of PASAD in practice, while achieving a reasonable time to detection. The goal of this experiment is to rationalize this choice by examining the behavior of the departure score over a relatively long period (roughly one month). The behavior of the departure score was monitored over a validation period spanning approximately 28 days (in simulation time). Then, it was observed that indeed there were but few false alarms as shown in Figure 11.

4.6 EXP. VI: Experimenting with Real Data

In the final experiment, we investigate the applicability and deployability of PASAD in a real-world setting. The data used in this experiment was collected from an operational water distribution plant. As described in §3.3, PASAD was run on limited-resource hardware along with packet-capturing, process-data extraction, and buffering mechanisms. Figure 12 displays the detection results of this experiment, where PASAD was tested on data extracted from a continuous register (left) and from an attribute register (right).

Table 1: PASAD's parameters for the experiments (see §2.9).

	Description	N	L	r
EXP. I	$\mathcal{SA}1$	10000	5000	26
	$\mathcal{SA}2$	1700	850	3
	$\mathcal{SA}3$	10000	5000	16
EXP. II	$\mathcal{DA}1$	10000	5000	16
	$\mathcal{DA}2$	10000	5000	43
EXP. III	**SWaT LIT301**	30000	5000	10
	SWaT AIT202	30000	5000	10
EXP. IV	**PASAD vs AR (1)**	2000	1000	12
	PASAD vs AR (2)	2400	1000	14
EXP. V	**False Alarms**	2000	1000	16
EXP. VI	**Real case (1)**	1000	500	13
	Real case (2)	50	15	9

PASAD is applicable to real-world scenarios. We applied PASAD on network traffic extracted from a real ICS, where our aim was to explore how well it would behave in a real-world setting. Therefore, we focused more on how effectively PASAD could handle irregularities in the data, and less on their actual causes. The results obtained in this experiment seem to confirm our observation about low false alarm rate. As can be seen in Figure 12, PASAD was tested over a period of roughly 106 days with data from a real system, which is considerably longer than what has previously been used in related research papers (e.g., 14 days in [22]), during which only few alarms were triggered. We argue that whether these alarms should be considered true or false depends on whether the operators are interested in knowing about such abrupt process changes as the ones shown in the figures.

5 RELATED WORK

The anomaly-based intrusion detection problem in the context of ICS has recently gained increasing attention. In a recent survey, Urbina et al. [51] presented a systematic literature review on physics-based attack detection in control systems. Two popular methods that were used by the papers they surveyed are the *Linear Dynamic State-space (LDS)* and the *Auto-Regressive (AR)* models.

System identification techniques can be used to create an LDS model of the physical process. In [51], an LDS model with a time delay was used to detect attacks on the water's pH level using the SWaT testbed [36]. Shoukry et al. [46] used the model together with a χ^2 anomaly detection statistic to build an authentication scheme (PyCRA) that they used to detect attacks on various kinds of active sensing subsystems. The same model was also used to detect spoofing attacks that can take control over an unmanned aircraft in [27], and to detect false data-injection attacks on state estimation in electric power grids in [35]. Cárdenas et al. [8] developed several attacks (Surge, Bias, and Geometric) on the TE process, and then used the LDS model together with a non-parametric CUSUM statistic for detection. Hadžiosmanović et al. [22] used the AR model, together with *Shewhart control limits*, to detect stream deviations in the process variables of operational water treatment plants.

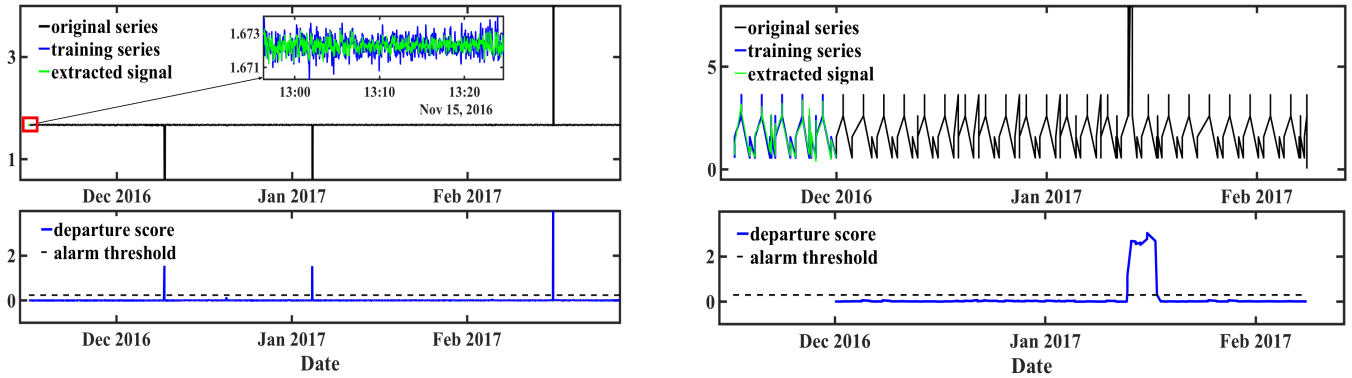

Figure 12: The behavior of PASAD while performing on 106-day-long process data, corresponding to a continuous register (left) and an attribute register (right), extracted from network traffic of an operational water distribution plant.

Approaches that use machine learning and data mining have been considered as well. Feng et al. [15] propose a multi-level anomaly-detection scheme where they combine a Bloom filter with a Long Short-Term Memory (LSTM) network to detect malicious traffic in a gas pipeline SCADA dataset. One-class classification techniques are evaluated on the same dataset in [40] where the authors consider Support Vector Data Description (SVDD) and kernel Principal Component Analysis (kPCA) for detecting intrusions in pipeline SCADA systems. Xiao et al. [54] use an LSTM neural network to detect malicious code executions on PLCs through a side-channel analysis of power consumption. Junejo and Goh [26] apply nine state-of-the-art machine-leaning classifiers to detect attacks on the SWaT testbed. Pan et al. [42] applied a so-called *common path* data mining technique on power system measurement data and audit logs that learns patterns and then classifies the system behavior over time into different scenarios. Clustering techniques are proposed in [28, 31] to detect attacks on the TE process targeting sensor measurements. Kiss et al. [28] considered the Gaussian mixture model to form sensor clusters and showed that the Gaussian model outperforms the k-means clustering algorithm for this particular problem. Krotofil et al. [31], on the other hand, considered an information-theoretic approach to form clusters of correlated sensors by discretizing sensor values to build discrete probability distributions and then using Shannon's entropy to perform implausibility checks on the sensor readings.

While LDS model-based techniques may accurately detect process misbehaviors, as stated in §1, they are difficult to build [15], and require a highly detailed description of the process that is not always available [28]. By contrast, PASAD does not require prior knowledge about the physical process as it learns the intrinsic dynamics of the system purely from historical sensor data. While machine learning methods do not require a model of the physical process, they involve a feature extraction and engineering phase, where *system-dependent* features need to be selected for training. Feature selection is tricky, hard to automate, and finding the best (most representative) features require a great deal of tuning and cross-validation. Moreover, the fact that features are constructed by combining various process variables and then transformed into high-dimensional feature spaces makes it difficult to identify the whereabouts of the attack and affects the interpretability of the detection results. By contrast, PASAD is specification-agnostic and uses easily available raw sensor data for training and monitoring.

Given that sensors in cyber-physical systems are intricately correlated and naturally fall into different clusters, we find that the work on identifying spoofed sensors using clustering techniques complements, rather than competes with, our approach.

To our knowledge, the only method in the related literature that monitors solely raw sensor measurements to detect attacks on control systems based on learned historical system dynamics is the AR method used in [22]. In §4.4, we presented a comparison with this method and showed that the results were in favor of PASAD. In particular, we showed that our method is less sensitive to noise and is thus able to detect covert attacks in sensor signals that are undetectable by AR.

6 CONCLUSION

Ensuring the security of cyber-physical systems requires the consideration of all aspects of their operation. What makes them fundamentally different from traditional IT systems is that they interact with the physical world. Accordingly, these systems are exposed to attacks that aim to cause tangible impact on the underlying physical process. We therefore acknowledged the importance of implementing intrusion detection capabilities at the process level as an advanced line of defense, and as a complementary measure to the classical solutions implemented elsewhere. We approached the problem by introducing a novel specification-agnostic technique that is capable of detecting stealthy attacks by monitoring time series of sensor measurements for structural changes in their behavior. We validated our approach by carrying out extensive tests on new carefully crafted attacks in a simulation setting using the popular TE model, a dataset generated by a physical ICS testbed, and on data from a real ICS.

ACKNOWLEDGMENTS

The research leading to these results has been supported by the Swedish Civil Contingencies Agency (MSB) through the project "RICS", and the Department of Economic Development and Infrastructures of the Basque Government through the project "CYBER-PREST" under Grant No.: KK-2018/00076.

REFERENCES

[1] Ali Abbasi and Majid Hashemi. 2016. Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. *Black Hat Europe* (2016).

[2] Marshall Abrams and Joe Weiss. 2008. Malicious Control System Cyber Security Attack Case Study—Maroochy Water Services, Australia. *McLean, VA: The MITRE Corporation* (2008).

[3] Matthew Allen and Carlo Pisani. 2018. Hacking and Cyber Warfare are Top Humanitarian Concerns. https://www.swissinfo.ch/eng/peter-maurer_hacking-and-cyber-warfare-are-top-humanitarian-concerns/43847744 . Last visited 2018-08-01.

[4] Magnus Almgren, Wissam Aoudi, Robert Gustafsson, Robin Krahl, and Andreas Lindhé. 2018. *The Nuts and Bolts of Deploying Process-Level IDS in Real Control Systems.* Technical Report. Chalmers University of Technology.

[5] Kaung Myat Aung. 2015. *Secure Water Treatment Testbed (SWaT): An Overview.* Technical Report. Singapore University of Technology and Design.

[6] George Box, Gwilym Jenkins, Gregory Reinsel, and Greta Ljung. 2015. *Time Series Analysis: Forecasting and Control.* John Wiley & Sons.

[7] David S Broomhead and Gregory P King. 1986. Extracting Qualitative Dynamics from Experimental Data. *Physica D: Nonlinear Phenomena* (1986).

[8] Alvaro Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. 2011. Attacks Against Process Control Systems: Risk Assessment, Detection, and Response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security.* ACM.

[9] Alvaro Cárdenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and Shankar Sastry. 2009. Challenges for Securing Cyber Physical Systems. In *Workshop on Future Directions in Cyber-Physical Systems Security.*

[10] Thomas Chen and Saeed Abu-Nimeh. 2011. Lessons from Stuxnet. *Computer* (2011).

[11] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. 2007. Using Model-Based Intrusion Detection for SCADA Networks. In *Proceedings of the SCADA security scientific symposium.* Citeseer.

[12] James Downs and Ernest Vogel. 1993. A Plant-Wide Industrial Process Control Problem. *Computers & Chemical Engineering* (1993).

[13] James B Elsner and Anastasios A Tsonis. 2013. *Singular Spectrum Analysis: A New Tool in Time Series Analysis.* Springer Science & Business Media.

[14] Nicolas Falliere, Liam Murchu, and Eric Chien. 2011. W32. Stuxnet Dossier. *White paper, Symantec Corp., Security Response* (2011).

[15] Cheng Feng, Tingting Li, and Deeph Chana. 2017. Multi-Level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks. In *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* IEEE.

[16] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2016. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *International Conference on Critical Information Infrastructures Security.* Springer.

[17] Nina Golyandina and Anton Korobeynikov. 2014. Basic Singular Spectrum Analysis and Forecasting with R. *Computational Statistics & Data Analysis* (2014).

[18] Nina Golyandina, Vladimir Viktorovich Nekrutkin, and Anatoly Alexandrovich Zhigljavsky. 2001. *Analysis of Time Series Structure: SSA and Related Techniques.* Chapman & Hall/CRC.

[19] Nina Golyandina and Anatoly Zhigljavsky. 2013. *Singular Spectrum Analysis for Time Series.* Springer Science & Business Media.

[20] Naman Govil, Anand Agrawal, and Nils Ole Tippenhauer. 2017. On Ladder Logic Bombs in Industrial Control Systems. In *Computer Security.* Springer.

[21] Bengt Gregory-Brown. 2017. Securing Industrial Control Systems-2017. *SANS Institute InfoSec Reading Room* (2017).

[22] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H Hartel. 2014. Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes. In *Proceedings of the 30th Annual Computer Security Applications Conference.* ACM.

[23] Hossein Hassani. 2010. A Brief Introduction to Singular Spectrum Analysis. *Optimal Decisions in Statistics and Data Analysis* (2010).

[24] John Hearon. 1967. Partially Isometric Matrices. *J. Res. Nat. Bur. Standards Sect. B* (1967).

[25] Blake Johnson, Dan Caban, Marina Krotofil, Dan Scali, Nathan Brubaker, and Christopher Glyer. 2017. Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure. https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html . Last visited 2018-08-01.

[26] Khurum Nazir Junejo and Jonathan Goh. 2016. Behaviour-Based Attack Detection and Classification in Cyber Physical Systems Using Machine Learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security.* ACM.

[27] Andrew Kerns, Daniel Shepard, Jahshan Bhatti, and Todd Humphreys. 2014. Unmanned Aircraft Capture and Control via GPS Spoofing. *Journal of Field Robotics* (2014).

[28] Istvan Kiss, Bela Genge, and Piroska Haller. 2015. A Clustering-Based Approach to Detect Cyber Attacks in Process Control Systems. In *Industrial Informatics (INDIN).*

[29] Marina Krotofil and Alvaro Cárdenas. 2013. Resilience of Process Control Systems to Cyber-Physical Attacks. In *Nordic Conference on Secure IT Systems.* Springer.

[30] Marina Krotofil and Jason Larsen. 2015. Rocking the Pocket Book: Hacking Chemical Plants. In *DefCon Conference, DEFCON.*

[31] Marina Krotofil, Jason Larson, and Dieter Gollmann. 2015. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15).* ACM.

[32] Truls Larsson, Kristin Hestetun, Espen Hovland, and Sigurd Skogestad. 2001. Self-Optimizing Control of a Large-Scale Plant: The Tennessee Eastman Process. *Industrial & Engineering Chemistry Research* (2001).

[33] Robert Lee, Michael Assante, and Tim Conway. 2014. *German Steel Mill Cyber Attack.* Technical Report. SANS Industrial Control Systems.

[34] Robert Lee, Michael Assante, and Tim Conway. 2016. *Analysis of the Cyber Attack on the Ukrainian Power Grid.* Technical Report. SANS Industrial Control Systems and E-ISAC.

[35] Yao Liu, Peng Ning, and Michael Reiter. 2011. False Data Injection Attacks Against State Estimation in Electric Power Grids. *ACM Transactions on Information and System Security (TISSEC)* (2011).

[36] Aditya Mathur and Nils Tippenhauer. 2016. SWaT: A Water Treatment Testbed for Research and Training on ICS Security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater).*

[37] Thomas McEvoy and Stephen Wolthusen. 2011. A Plant-Wide Industrial Process Control Security Problem. In *International Conference on Critical Infrastructure Protection.* Springer.

[38] Yilin Mo and Bruno Sinopoli. 2016. On the Performance Degradation of Cyber-Physical Systems under Stealthy Integrity Attacks. *IEEE Trans. Automat. Control* (2016).

[39] Valentina Moskvina and Anatoly Zhigljavsky. 2003. An Algorithm Based on Singular Spectrum Analysis for Change-Point Detection. *Communications in Statistics-Simulation and Computation* (2003).

[40] Patric Nader, Paul Honeine, and Pierre Beauseroy. 2014. Lp-Norms in One-Class Classification for Intrusion Detection in SCADA Systems. *IEEE Transactions on Industrial Informatics* (2014).

[41] Nell Nelson. 2016. The Impact of Dragonfly Malware on Industrial Control Systems. *SANS Institute* (2016).

[42] Shengyi Pan, Thomas Morris, and Uttam Adhikari. 2015. Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems. *IEEE Transactions on Smart Grid* (2015).

[43] Vern Paxson. 1999. Bro: A System for Detecting Network Intruders in Real-Time. *Computer networks* (1999).

[44] Pavel Polityuk, Oleg Vukmanovic, and Stephen Jewkes. 2017. Ukraine's Power Outage was a Cyber Attack: Ukrenergo. https://www.reuters.com/article/us-ukraine-cyber-attack-energy/ukraines-power-outage-was-a-cyber-attack-ukrenergo-idUSKBN1521BA . Last visited 2018-08-01.

[45] Lawrence Ricker. 1996. Decentralized Control of the Tennessee Eastman Challenge Process. *Journal of Process Control* (1996).

[46] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. 2015. PyCRA: Physical Challenge-Response Authentication for Active Sensors under Spoofing Attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* ACM.

[47] Ralf Spenneberg, Maik Brüggemann, and Hendrik Schwartke. 2016. PLC-Blaster: A Worm Living Solely in the PLC. *Black Hat Asia, Marina Bay Sands, Singapore* (2016).

[48] Keith Stouffer, Joe Falco, and Karen Scarfone. 2011. Guide to Industrial Control Systems (ICS) Security. *NIST special publication* (2011).

[49] Gilbert Strang. 2016. *Introduction to Linear Algebra.* Wellesley-Cambridge Press.

[50] David Urbina, Jairo Giraldo, Alvaro Cárdenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.* ACM.

[51] David Urbina, Jairo Giraldo, Alvaro Cárdenas, Junia Valente, Mustafa Faisal, Nils Ole Tippenhauer, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. *Survey and New Directions for Physics-Based Attack Detection in Control Systems.* Technical Report. National Institute of Standards and Technology.

[52] Robert Vautard and Michael Ghil. 1989. Singular Spectrum Analysis in Nonlinear Dynamics, with Applications to Paleoclimatic Time Series. *Physica D: Nonlinear Phenomena* (1989).

[53] Oleg Vukmanovic and Stephen Jewkes. 2017. Suspected Russia-Backed Hackers Target Baltic Energy Networks. http://mobile.reuters.com/article/idUSKBN1871W5 . Last visited 2018-08-01.

[54] Yu-jun Xiao, Wen-yuan Xu, Zhen-hua Jia, Zhuo-ran Ma, and Dong-lian Qi. 2017. NIPAD: A Non-Invasive Power-Based Anomaly Detection Scheme for Programmable Logic Controllers. *Frontiers of Information Technology & Electronic Engineering* (2017).

APPENDICES

A

Proof that $||\mathbf{U}^T\mathbf{P}\mathbf{x}|| = ||\mathbf{P}\mathbf{x}||$

Since $\mathbf{U}^T\mathbf{P} = \mathbf{U}^T$ from (11), and $\mathbf{P} = \mathbf{U}\mathbf{U}^T$ from (1), what remains to show is that

$$||\mathbf{U}^T\mathbf{x}|| = ||\mathbf{U}\mathbf{U}^T\mathbf{x}||.$$

Let $\mathbf{x}$ be any vector in $\mathbb{R}^L$. Noting that the square Euclidean norm is equal to the dot product of a vector with itself, we write

$$\begin{aligned}
||\mathbf{U}\mathbf{U}^T\mathbf{x}||^2 &= \mathbf{U}\mathbf{U}^T\mathbf{x} \cdot \mathbf{U}\mathbf{U}^T\mathbf{x} \\
&= (\mathbf{U}\mathbf{U}^T\mathbf{x})^T(\mathbf{U}\mathbf{U}^T\mathbf{x}) \qquad (\cdot) \text{ in matrix notation} \\
&= \mathbf{x}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})\mathbf{U}^T\mathbf{x} \\
&= \mathbf{x}^T\mathbf{U}\mathbf{U}^T\mathbf{x} \qquad \text{since } \mathbf{U}^T\mathbf{U} = \mathbf{I} \\
&= (\mathbf{U}\mathbf{x})^T(\mathbf{U}\mathbf{x}) = ||\mathbf{U}^T\mathbf{x}||^2.
\end{aligned}$$

Therefore, $||\mathbf{U}\mathbf{U}^T\mathbf{x}|| = ||\mathbf{U}^T\mathbf{x}|| \; \forall \mathbf{x} \in \mathbb{R}^L$.

B

Proof that $\mathcal{R}(\mathbf{U}) = \mathcal{R}(\mathbf{P})$

First note that since all columns of $\mathbf{U}$ are linearly independent, $\mathcal{K}(\mathbf{U}) = \{\mathbf{0}\}$, which means that $\forall\, \mathbf{y} \in \mathcal{K}(\mathbf{U})$,

$$\mathbf{U}\mathbf{y} = 0 \iff \mathbf{y} = 0. \tag{15}$$

We also have $\mathcal{K}(\mathbf{P}) = \mathcal{K}(\mathbf{U}\mathbf{U}^T) = \mathcal{K}(\mathbf{U}^T)$ since

$$\begin{aligned}
\mathbf{x} \in \mathcal{K}(\mathbf{U}\mathbf{U}^T) &\iff \mathbf{U}\mathbf{U}^T\mathbf{x} = 0 \qquad \text{(by definition)} \\
&\iff \mathbf{U}^T\mathbf{x} = 0 \qquad \text{(by (15))} \qquad (16) \\
&\iff \mathbf{x} \in \mathcal{K}(\mathbf{U}^T).
\end{aligned}$$

Finally,

$$\begin{aligned}
\mathbf{x} \in \mathcal{R}(\mathbf{U}\mathbf{U}^T) &\iff \mathbf{x} \perp \mathcal{K}(\mathbf{U}\mathbf{U}^T) \quad \text{(by (1))} \\
&\iff \mathbf{x} \perp \mathcal{K}(\mathbf{U}^T) \quad \text{(by (16))} \\
&\iff \mathbf{x} \in \mathcal{R}(\mathbf{U}).
\end{aligned}$$

Therefore, $\mathcal{R}(\mathbf{U}) = \mathcal{R}(\mathbf{U}\mathbf{U}^T) = \mathcal{R}(\mathbf{P})$.

C

Proof that $\tilde{\mathbf{U}} : \mathcal{R}(\mathbf{U}) \to \mathcal{R}(\mathbf{U}^T)$ is an Isometric Isomorphism

Recall that $\tilde{\mathbf{U}}$ is the restriction of $\mathbf{U}^\mathrm{T}$ to the orthogonal complement of its kernel $\mathcal{K}(\mathbf{U}^T)^\perp = \mathcal{R}(\mathbf{U})$; i.e., $\tilde{\mathbf{U}} = \mathbf{U}^T\big|_{\mathcal{R}(\mathbf{U})}$. Since $\tilde{\mathbf{U}} : \mathcal{R}(\mathbf{U}) \to \mathcal{R}(\mathbf{U}^T)$ is a *linear map*, by definition it satisfies the following two conditions:

$$\begin{aligned}
&\tilde{\mathbf{U}}(\mathbf{v} + \mathbf{v}') = \tilde{\mathbf{U}}(\mathbf{v}) + \tilde{\mathbf{U}}(\mathbf{v}') && \forall\, \mathbf{v}, \mathbf{v}' \in \mathcal{R}(\mathbf{U}) \\
&\tilde{\mathbf{U}}(c\mathbf{v}) = c\tilde{\mathbf{U}}\mathbf{v} && \forall\, \mathbf{v} \in \mathcal{R}(\mathbf{U}), c \in \mathbb{R}.
\end{aligned}$$

Hence, $\tilde{\mathbf{U}}$ is a *homomorphism*. For $\tilde{\mathbf{U}}$ to be an *isomorphism*, it must further be bijective, i.e., both injective and surjective.
To show that $\tilde{\mathbf{U}}$ is injective (one-to-one), let $\mathbf{v}, \mathbf{v}' \in \mathcal{R}(\mathbf{U})$ have the same image $\mathbf{w} \in \mathcal{R}(\mathbf{U}^T)$, then it must be that $\mathbf{v} = \mathbf{v}'$.
Since $\mathbf{w} \in \mathcal{R}(\mathbf{U}^T)$ is the image of both $\mathbf{v}$ and $\mathbf{v}'$ we have

$$\begin{aligned}
\mathbf{w} &= \tilde{\mathbf{U}}\mathbf{v} = \tilde{\mathbf{U}}\mathbf{v}' \\
&\Rightarrow \tilde{\mathbf{U}}\mathbf{v} - \tilde{\mathbf{U}}\mathbf{v}' = 0 \\
&\Rightarrow \mathbf{U}^T\mathbf{v} - \mathbf{U}^T\mathbf{v}' = 0 \\
&\Rightarrow \mathbf{U}^T(\mathbf{v} - \mathbf{v}') = 0 \\
&\Rightarrow \mathbf{v} - \mathbf{v}' \in \mathcal{K}(\mathbf{U}^T)
\end{aligned}$$

But $\mathbf{v} - \mathbf{v}' \in \mathcal{R}(\mathbf{U})$, hence $\mathbf{v} - \mathbf{v}' \in \mathcal{R}(\mathbf{U}) \cap \mathcal{K}(\mathbf{U}^T) \Rightarrow \mathbf{v} - \mathbf{v}' \in \mathcal{R}(\mathbf{U}) \cap \mathcal{R}(\mathbf{U})^\perp = \{\mathbf{0}\} \Rightarrow \mathbf{v} - \mathbf{v}' = \mathbf{0}$. Thus $\mathbf{v} = \mathbf{v}'$.
To see that $\tilde{\mathbf{U}}$ is surjective, note that for every $\mathbf{w} \in \mathcal{R}(\mathbf{U})$, there exists $\mathbf{v} = \tilde{\mathbf{U}}^+\mathbf{w} \in \mathcal{R}(\mathbf{U}^T)$ such that $\tilde{\mathbf{U}}\mathbf{v} = \tilde{\mathbf{U}}\tilde{\mathbf{U}}^+\mathbf{w} = \mathbf{U}^T\mathbf{U}\mathbf{w} = \mathbf{w}$.
Therefore $\tilde{\mathbf{U}}$ is a linear bijective homomorphism, or equivalently, a linear isomorphism.